Edge Crossing Minimization in Graphs - A Survey

Aftab Hussain, Md. Saidur Rahman

Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka - 1000, Bangladesh.
get_aftab2003@yahoo.com, saidurrahman@cse.buet.ac.bd

Abstract. In this survey paper we study the various problems on edge crossing minimization in graphs. Different crossing minimization algorithms have been adopted for different graph models as per the requirements of their respective application areas. Here we elicit on such algorithms. A significant portion of this paper is also dedicated to the study of minimizing crossing in graphs that are used to represent public transportation maps. We also give some future directions for further research in this field. Additionally, we propose a result for bipartite graphs.

1 Introduction

With the increasing application of graph drawing in different areas of science, technology and engineering, the problem of minimizing edge crossings in graphs has become an important issue of graph drawing. The reason for its growing importance has mainly been due to the need for better visualization. Graphs with a large number of crossings create cluttering making them difficult to read and understand. For instance, this problem is faced in public transportation maps and circuitry diagrams. In addition to that, edge crossings become a nuisance in designing electrical circuit diagrams and VLSI circuit layouts, where a larger number of crossings would increase the cost of the design [10]. Thus, crossing minimization is a vital issue, having great application−specific significance.

However, determining the minimum number of crossing has not been easy. Early explorations in this field did yield some methods but were computationally very expensive and were not scalable with large graphs. An example is Nicholson's permutation procedure for minimizing the number of crossings in a network, [14], which is discussed in Section 2. The problem was in fact proved to be NP-hard by Masuda et Al. when a certain set of conditions were imposed on the graph [12]. They named the problem as the *fixed linear crossing number problem* which they define as the problem of finding a linear embedding of a graph with the minimum number of edge crossings under a specified vertex ordering. Their findings are also discussed in Section 2. Another interesting example is the work done by Shen Wei−xiang and Huang Jing−wei [17]. They presented an edge crossing minimization algorithm for hierarchical graphs based on genetic algorithm. Experimentally their algorithm yielded some sound results but consumed considerable time to run.

Nonetheless, there were significant achievements in this area for some classes of graphs. Sykora et Al. [16], proposed algorithms and methods for k-planar drawings of general graphs together with lower bound techniques. Christoph Buchheim and Seok-Hee Hong [6] devised an $\mathrm{O}(m \log m)$ algorithm for computing a crossing minimal drawing for a graphs with a given symmetry.

A particularly interesting area where there were many useful findings with respect to the edge crossing minimization problem is public transportation maps. Public transportation maps pose a scenario where graphs have a fixed embedding, where vertices represent stations and the edges represent the tracks (in case of metro maps for example) or roads. Then our objective becomes to remove those crossings from our graph that are avoidable. For example, Fig. 1(a) depicts a station (represented in a map) with 2 tracks, both entering the station from the same direction and leaving it and heading towards the same direction. That being the case, the crossing becomes avoidable and thus the lines can be rearranged as shown in Fig. 1(b).

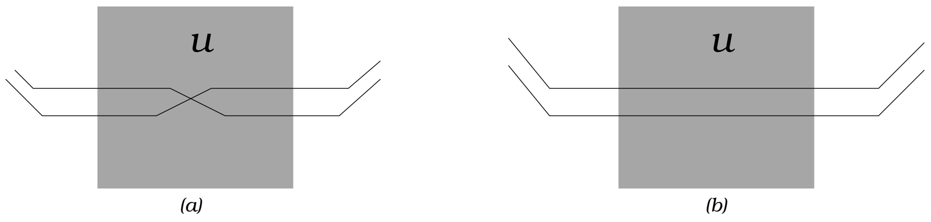


Fig. 1. An avoidable crossing at a station.

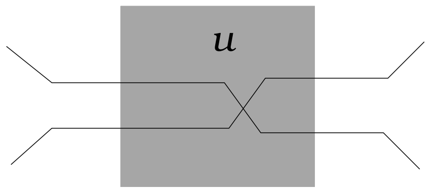


Fig. 2. An unavoidable crossing at a station.

However, for Fig. 2, we see the 2 lines (tracks) both enter station u from different directions and leave it heading for different directions. In that case the crossing in station u becomes unavoidable. (It is to be noted that the figures represent graphs with a fixed embedding; the stations are represented as polygons and the tracks pass through the station edges. This type of model was first introduced by Benkert et. Al. [5] a detailed discussion on these graph models and the ways of minimizing edge crossing in those graphs is given in Section 3.)

We organize our survey paper as follows: In Section 2, we discuss various algorithms that have been devised for an assortment of graphs. In Section 3, we discuss works that have been done on the *Metro Line Crossing Minimization problem*. In the penultimate section, Section 4, we suggest opportunities for further development in the study of edge crossing minimization, and give a result for bipartite graphs. We conclude our paper in Section 5.

2 Edge crossing minimization in different types of graphs

In this section we consider edge crossing minimization problems for different classes of graphs.

2.1 Crossing minimization in Book Drawings of graphs

A book drawing of a graph $G = (V, E)$ is a drawing of G where all vertices of set V are positioned on a straight line and all the edges of edge set E are drawn as semicircles above and below the straight line. Such a graph is similar to a book, where the vertices on the straight line represent the backbone of a book and edges in the half-planes represent the pages of the book [10]. Any graph can be transformed into a book drawing [14] and the problem of minimizing the number of edge crossings can then be addressed by analyzing the book drawing. Different studies and techniques have emerged on this aspect of book drawings. An example of a book drawing of a graph (with numbered vertices) is shown in Fig 3

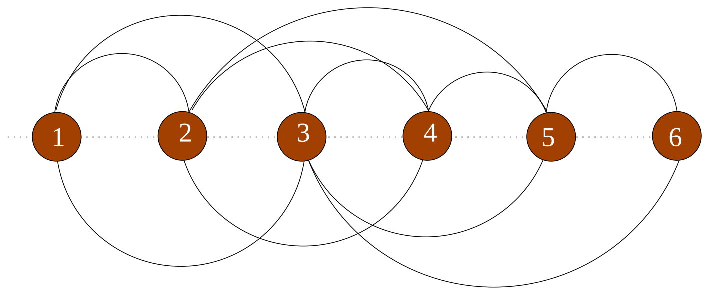


Fig. 3. A book drawing of a graph

2.1.1 A Permutation Technique

We shall discuss a permutation technique by Nicholson [14] to find the minimum number of edge crossings. The assumptions are: the relative positions of the vertices are unrestricted, and only simple graphs are considered, i.e., there will be at most one edge between any 2 vertices. The objective of the technique is to find a positioning of the vertices on the straight line that will yield a minimum number of crossings in the entire graph. For convenience, the straight line is taken to be horizontal.

The positions of the vertices are denoted by their place in the sequence of vertices along the straight line. Let $p(j)$ denote the vertex number occupying the jth position along the straight line, so that if there are N vertices, the vertex arrangement is expressed by a permutation $[P]$ of size N. Let there be M edges between the vertices and denote by $q(m)$ the route of the mth edge. Let $q(m) = 1$ if the mth edge is a semicircle above the node line and let $q(m)$=-1 if it lies below the straight horizontal line. Then the routing of the edges can be expressed by a permutation $[Q]$ of size M in which all the elements are +1 or -1. Therefore the layout of the graph can be represented by a joint permutation $[P;Q]=[p(1),p(2), \ldots, p(N);q(1),q(2), \ldots, q(M)]$ where $[P]$ is any permutation of the first N integers and $q(j) = +1$ or -1 for j=1.

Now we need to evaluate the number of crossings which will occur for any given permutation $[P,Q]$. Let $\{c(m),c'(m)\}$ for m=1,M be the list of M edges where the edge $\{c(m),c'(m)\}$ joins the vertex numbers $c(m)$ and $c'(m)$. 2 auxiliary matrices A and B of dimensions $N \times N$ are used, such that:

$A(i,j)$=1, if $\{c(m),c'(m)\} \equiv \{p(i),p(j)\}$ and $q(m) = 1$
for some m=0
=0 otherwise.

$B(i,j)$=1, if $\{c(m),c'(m)\} \equiv \{p(i),p(j)\}$ and $q(m) =$ -1
for some m=0
=0 otherwise.

Thus A and B are the upper and lower edge matrices, respectively. The number of crossings associated with any permutation $[P,Q]$ is thus expressed as,

$$F[P,Q] = \sum_{i=1}^{N-3} \sum_{i=1}^{N-1} \Big\{ A(i,j) \sum_{k=i+1}^{j-1} \sum_{l=j+1}^{N} A(k,1) + B(i,j) \sum_{k=i+1}^{j-1} \sum_{l=j+1}^{N} B(k,1) \Big\}$$

Now, one can find the number of crossings for each of the permutation and determine the permutation with the lowest number of crossings. Clearly, this is not feasible, as one will have to search through all possible permutations $((N-1)!/2)$. To address this issue, Nicholson [14] reduces the field of search by constructing an optimal permutation ,based on the formula(above), for the number of crossings, targetting a certain level of optimality. In experiments, the permutation technique took long execution times for large graphs. Nicholson proposed that the large graph might be partitioned into a set of smaller graphs; the permutation method could then be applied to each of the smaller graphs in turn and subsequently to the overall graph, keeping their internal relationships fixed.

2.1.2 Heuristics for 1-page and 2-page drawings

1-page and 2-page drawings are two variants of book drawings. A *one-page drawing* of a connected graph $G = (V, E)$ with n vertices m edges is a drawing the vertices are placed along a circle, and the edges are drawn as straight lines [10]. Therefore edge crossing minimization for such graphs can be reduced to the task of finding an ordering $f : V \to \{0, 1, ...n-1\}$ of the vertices such that the number of edge crossings can be minimized. (The minimial possible number of edge crossings in this drawing is called the one-page crossing number.) This problem was proved to be NP-hard [11]. A *two-page drawing* of graph G is a drawing of G where the vertices are placed along a circle and every edge is completely drawn in one of two colours. The two-page crossing number of a graph G is the smallest possible number of crossings of edges of the same colour. This problem was also proven to be NP-hard [12]. Nonetheless, studying these two types of drawings is useful as they provide upper bounds for standard planar crossing number. We present several such heuristics. In each of these heuristic algorithms the approach is based on the step-by-step placement of vertices and edges in manner that will yield a low number of edge crossings.

Algorithm of Baur and Brandes [4]. This algorithm works for one-page drawings. Their algorithm consists of two phases; a greedy phase and a shifting phase. In the greedy phase, in each step a vertex with the largest number of already placed neighbours is selected. If there exists more than one such vertex, vertices with fewer unplaced neighbours are selected. The selected vertex is then appended to the end that yields fewer crossings of edges being closed with open edges. This phase has a time complexity of $O((n+m)logn)$, where n is the number of edges and m is the number of vertices. In the shifting phase, every vertex is moved along a fixed ordering of all other vertices. The vertex is then placed in its (locally) optimal position. This phase's complexity is $O(nm)$.

AVSDF+ algorithm [9]. This is another algorithm for one-page drawings, that works in 2 phases, greedy and adjusting. In the greedy phase we first place the vertex with the smallest degree. Then we visit the adjacencies of the current vertex, which have not been visited yet, such that the smallest degree vertexhas the highest priority for visiting. This phase takes $O(m)$ time. In the next phase (adjusting), a vertex with the largest number of crossings caused by its incident edges is selected. Then the best position among the current one and the ones next to its adjacent vertices is found. The procedure is repeated until no vertex can be adjusted. The time complexity of this phase is $O(mn)$.

A Hybrid heuristic algorithm [10]. This algorithm works for two-page drawings. It is a hybrid of the previous technique and another heuristic algorithm for 2-page drawing. It places an edge incident to the currently placed vertex based on the smallest crossings produced by the edge and already placed edges. Edges of a graph are distributed on one of two pages according to their slopes. If the angle between an edge and the horizontal axis is larger than $90\,^\circ$, the edge is put on page 2, otherwise the edge is put on page 1. The runtime of this algorithm is $O(mn)$.

Length-based edge placement strategy [7]. The principle of this strategy for 2-page drawings is that, the longer length of an edge is, the larger is the probability of its crossing with other edges. Edges of length=1 are assumed to create no crossings. a sorted edge list is thus created without the edges of length=1, on non-increasing length. Each edge is then placed in turn . The process is iterated until there is no improvement or 5 iterations were done (Cimikowski [7] found 5 times to be sufficient experimentally). The running time is $O(m)$.

Edge adjustment according to descending number of crossings [10]. This 2-page drawings strategy is based on an initial one-page drawing. Edges which create most crossings are put it to the other page. The number of crossings for each edge are then recalculated, which is in turn used to sort the edges. The process is repeated until no edge can be adjusted. The best case and worst case running times are $O(m))$ and $O(m^2)$, respectively.

2.1.3 Masuda et. Al's findings

Their work deals with crossing minimization in linear embeddings of graphs [12] in other words, *book embeddings* of graphs. They dealt with both simple and multigraphs. They give the following definitions and proofs:
Crossing number problem. The problem of determining, for a given integer K, whether a graph G can be embedded in the plane with K or fewer pairwise crossings of the edges (not including the intersections of the edges at their common endpoints).
Crossing minimization problem. The problem of embedding a graph in the plane with the minimum number of edge crossings.
Fixed linear crossing minimization problem. The problem of finding a linear embedding of a graph with the minimum number of edge crossings under a specified vertex ordering. We call the problem of finding such an embedding with no vertex ordering specified the *free linear crossing minimization problem.* Furthermore, we define the free linear crossing number problem to be that of determining, for a given integer K, whether there is a linear embedding of a graph with K or fewer edge-crossings. (Algorithms for addressing the free linear crossing number problem have already been discussed in subsections **2.1.1** and **2.1.2**.)

Let $f : V \to (1, 2, ..., |V|)$ be a one-to-one function. We call an embedding G' of G in the plane an *f-fixed linear ernbedding*, or simply an f-linear embedding, if
1. each vertex $v\epsilon V$ is placed on the x-axis l with x-coordinate $f(v)$.
2. the edges E are drawn by semi-ellipses, and

3. the semi-ellipses for nonparallel edges intersect in at most one point.

FIXED LINEAR CROSSING NUMBER:
Instance : Graph $G = (V, E)$, integer $K \geq 0$ and one-to-one function $f : V \rightarrow \{1, 2, \ldots, |V|\}$.
Question : $v_f(G) \leq K$?

They denote $v_f(G)$ as the least total number of crossings among all f-linear embeddings of G. They proved that the fixed linear crossing number problem is $NP-$hard. In order to prove this they proved the following theorems and lemmas.

Theorem 1. *The fixed linear crossing number problem is NP-complete.*

They prove this theorem by using the set splitting problem, which is known to be NP-complete, formally defined as follows:

SET SPLITTING:
Instance : Collection $S = \{S_1, S_2, \ldots, S_n\}$ of subsets of a finite set $X = \{x_1, x_2, \ldots, x_m\}$.
Question : Is there a set splitting of X with respect to S, i.e, a partition of X into two subsets X_1 and X_2 such that no subset in S is entirely contained in either X_1 or X_2?

They base their proof of the theorem on the following lemmas,

Lemma 1. *G, f, and K can be constructed from S and X in polynomial time with respect to m and n.*

Lemma 2. *The following 2 statements are equivalent:*
1. *X has a set splitting with respect to S.*
2. *$v_f(G) \leq K$*

Lemma 3. *If Statement 2 holds for G, f, and K, then Statement 1 holds for S and X.*

Theorem 2. *The fixed linear crossing number problem remains NP-complete even if a given graph is simple and each of its connected components is a single edge.*

They prove this theorem by converting a multi graph to a simple graph by splitting each vertex v in $d(v)$ vertices corresponding to the incident edges where $d(v)$ denotes the degree of v in G and henceforth showing that the two graphs are in one-to-one correspondence that preserves crossings.

2.2 Crossing Minimisation in Symmetric graphs

Symmetric graphs produce aesthetically pleasing graphs and thus has been a key area of study. Christoph Buchheim and S. Hong [6] were the first to introduce the problem of crossing minimisation for symmetries. Before presenting some of their findings we give some preliminary ideas.

A *symmetry* or *geometric automorphism* of a graph G is a permutation of the vertices of G that is induced by a symmetric drawing of G, i.e., by a drawing of G that is fixed by some non-trivial isometry of the plane. There are two different types of symmetries [8]: a $reflection$ or *axial symmetry* is a symmetry induced by a reflection of the plane at an axis, the reflection axis; a *rotation* is a symmetry induced by a rotation of the plane around a center point, the rotation center. The order of a symmetry π is the smallest positive integer k such that πk equals the identity. In particular, all reflections have order one or two. For a vertex v, the orbit (or cycle) of v under π consists of the set of nodes $\pi^k(v)$ for all integers k. Analogously, the orbit of an edge $e = (v, w)$ is defined as the set of all edges $\pi^k(e) = (\pi^k(v), \pi^k(w))$. The vertex and edge orbits define equivalence relations on the sets of nodes and edges, respectively. An edge is called intra-orbit if the vertices connected by this edge belong to the same node orbit, otherwise it is called an inter-orbit. A node v of G is fixed by π if $\pi(v) = v$. The *orbit graph* G/π of π is the graph resulting from G by identifying each vertex v with its image $\pi(v)$ and deleting multiple edges and loops. Hence every vertex of G/π represents a π-orbit of G and every edge of G/π corresponds to a set of orbits of inter-orbit edges of G; see Figure. 4.

In a drawing of a rotation, all nodes of a common orbit have the same distance to the rotation center; the corresponding circle is called the *orbit circle.* For a drawing of a reflection, the nodes of a common orbit lie on the same line orthogonal to the reflection axis; this line is called the *orbit line.*

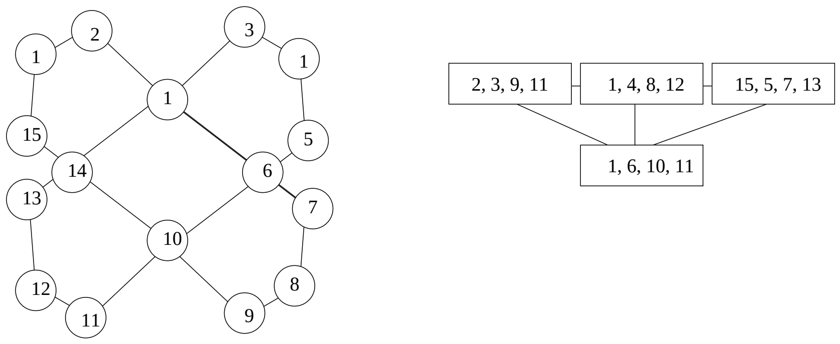


Fig. 4. A rotation and its orbit graph

A graph is *planar* if it has a *plane drawing*, i.e., a two-dimensional drawing without edge crossings. A symmetry π of a graph G is planar if there is a drawing of π that is planar as a drawing of G. We call a drawing of a symmetry *loopless* if all orbit circles or lines are distinct and if no edge crosses more orbit circles or lines than necessary. More precisely, in a loopless drawing of a rotation, every curve connecting two orbits may only cross the circles of the orbits in between, and each only once. Analogously, in a loopless drawing of a reflection, every curve connecting two orbits may only cross the lines of orbits in between, and each only once. In a loopless drawing, we add a *dummy node* at each crossing of an edge with an orbit circle or line; see Figure 5.

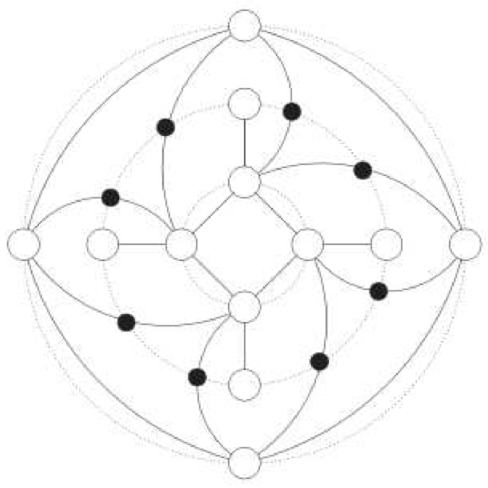


Fig. 5. A rotation and its orbit graph

We now give the problems that were dealt with in Buchheim's paper [6].

Problem (SCM). Given a symmetry π, compute a drawing of π with a minimal number of edge crossings.

Problem (SCM+). Given a symmetry π , compute a loopless drawing of π with a minimal number of edge crossings.

Problem (SCM+1). Given a symmetry π and a fixed order of its orbits, compute a loopless drawing of π respecting this order such that the number of edge crossings in this drawing is minimal.

Problem (SCM+2). Given a symmetry π, a fixed order of its orbits, and a fixed order of nodes and dummy nodes on each orbit circle or orbit line, compute a loopless drawing of π respecting these orders such that the number of edge crossings in this drawing is minimal.

Based on the following lemmas they give some theorems on the above mentioned problems.

Lemma 4. *For every planar drawing of a symmetry π , there exists a planar loopless drawing of π inducing the same embedding of the underlying graph.*

Lemma 5. *The problem (SCM) can be reduced to (SCM+) in $O(m^2)$ time, also if both problems are restricted to reflections or rotations of fixed order.*

Here are the theorems.

Theorem 3. *The problems (SCM) and (SCM+) are NP-hard, even if restricted to reflections or rotations of fixed order.*

Theorem 4. *The problem (SCM+1) is NP-hard, even if restricted to reflections or rotations of fixed order.*

Theorem 5. *The problem (SCM+2) can be solved in $O(mlogm)$ time for symmetries without dummy nodes. The corresponding number of edge crossings can be computed in $O(m/k \times logm)$ time, where k is the order of the symmetry.*

Corollary 1. *The problem (SCM+) can be solved in $O(mlogm)$ time if the orbit graph of π is a path and no dummy nodes are allowed.*

Corollary 2. *The problem (SCM) can be solved in $O(mlogm)$ time for symmetries without inter-orbit edges.*

2.3 Edge Crossing Minimization for Hierarchical graphs

In this section we discuss the technique used by Shen Wei-xiang and Huange Jing-wei [17]. Their method is based on genetic algorithms and has a simple implementation. They showed that the technique is efficient based on computational experiments. They mainly aimed at edge crossing minimization for layered digraphs. A k-layered digraph is a directed graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, which satisfy the follow- ing properties:

1) V is partitioned into k nonempty subsets, $V = V_1 \cup V_2 \cup \ldots \cup V_k, (V_i \cap V_j = \oslash, i \neq j)$ where $V_i(i = 1, 2, \ldots, k)$ is called the $i - th$ layer, k is the height of the layered digraph.

2) If edge $(u, v) \in E$,where $u \in V andv \in V$, then $i \leq$. A k-layered digraph as $G = (V, E, k)$.

In a layered drawing the number of crossings (CN) can be optimized by reordering the vertices, as shown in Figure 6. Since the direction of the edges have no effect on crossings, the graph is considered to be undirected in the algorithm.

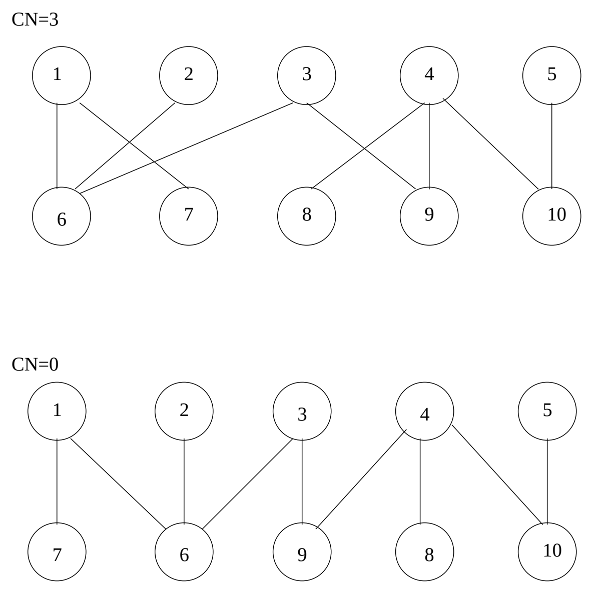


Fig. 6. 2 drawings of a layered digraph

Given a two-layered digraph $G = (V, E, 2)$, where $V = V_1 \cup V_2 and V_1 \cap V_2 = \oslash$, let $n_1 = |V_1|, n_2 = |V_2|, m = |E|$, and let $N(v) = \{w \in V|(v, w) \in E\}$ denote the set of neighbors of $v \in V$. Let $V_1 = \{x_1, x_2, \ldots, x_{n_1}\}, V_2 = \{y_1, y_2, \ldots, y_{n_1}\}$, and π_i, be an ordering of the vertices in layer $V_i (i = 1, 2)$, where $\pi_{i(v)}$ is the position of vertex x_v in layer V_i, then any solution is completely specified by the ordering π_1, π_2 of V_1, V_2. For $k = 1, 2$ let $\delta^k_{ij} = 1$ if $I\pi_k(i) < \pi_k(j)$ and 0 otherwise. Then given π_1, π_2 the number of crossing is, $C(\pi_1, \pi_2) = C(\delta^1, \delta^2) =$

$$\sum_{i=1}^{n_2-1} \sum_{j=1+1}^{n_2} \sum_{k \in N(i)} \sum_{l \in N(j)} (\delta^1_{kl} \cdot \delta^2_{ji} + \delta^1_{lk} \cdot \delta^2_{ij}) \cdot$$
$$\sum_{k=1}^{n_1-1} \sum_{l=k+1}^{n_1} \sum_{i \in N(k)} \sum_{j \in N(j)} (\delta^1_{kl} \cdot \delta^2_{ji} + \delta^1_{lk} \cdot \delta^2_{ij}) \text{ ....(1)}$$

Formula 1 can be used for k-layered digraphs to calculate the number of crossings by repeatedly using it $k - 1$ times from the top to the bottom of the layered digraph.

Genetic algorithms are easy to escape local optimal and find the true optimal solution. According to the above discussion, the objective function can be designed as $f(\pi_1, \pi_2, \ldots \pi_k) = CN$. In this way, the original problem is translated into a an optimization problem as $\text{Min} f(\pi_1, \pi_2, \ldots \pi_k)$, where $\pi_1, \pi_2, \ldots \pi_k$ are feasible orderings of each layer. The steps that were used to design the genetic algorithm: Coding, design of fitness function, design of selection operator, specification of control parameter, design of genetic operators, and specification of termination conditions. (Details of the steps have been omitted.)

For the coding part, they used δ_{ij}^k for vertices of each layer. This coding preserves both the inter and the local permutation information for each layer. For design the fitness function, they deduced a formula for the objective function f, and converted it into the fitness function F by inversing f. They adopted $(\mu + \lambda)$-Selection Operator in ex- pansive sampling space, where the sampling space is composed of father and son chromosomes. According to this operator, father μ and son λ chromosomes compete for survival, and finally, the best father and son compose next population. As for the control parameters, they set the population scale to 40 and crossover probability to 0. 9, mutating at a rate of 0.4. They use a uniform crossover operator and a simple mutation operator. They terminate their algorithm after a predetermined number of runs.

Their algorithm was found to strong and adaptive to different applications (graphs), with the requirement of designing different object functions for each application. However the algorithm had a very long runtime.

2.4 Crossing numbers of general graphs from $k-$ planar drawings

Here we present the findings of Sykora et. Al. [16]. They give general bounds for the k-planar crossing number and exposes an important extremal problem: how does $cr_k(G)$ decrease when k increases. It is thus important to understand the meaning k-planar crossing number and some other relevant terms.

Let $cr(G)$ denote the standard crossing number of a graph G, i.e. the minimum number of crossings of its edges over all possible drawings of G in the plane. For $k \geq 2$, define the *k-planar crossing number* is defined as,

$$cr_k(G) = min\{cr(G_1) + cr(G_2) + \ldots + cr(G_k)\},$$

where the minimum is taken over all edge disjoint subgraphs $G_i = (V, E_i), i = 1, 2, ..., k$, so that $E = E_1 \cup E_2 \cup \ldots \cup E_k$.
The problem can be viewed as a drawing of edges of G in k planes with minimum number of crossings. One can easily see that a k-planar drawing can be redrawn in such a way that any vertex can be placed on the same place in each plane without changing the number of crossings.
Thickness. The thickness $\theta(G)$ of G is the minimum number of planar graphs whose union is G. By definition, $cr_k(G) = 0$ if and only if $\theta(G) \leq k$.
Arboricity. Arboricity, $a(G)$, of G, is the minimum number of acyclic subgraphs whose union covers E. By a well-known theorem of Nash-Williams [13],

$$a(G) = max_{H \subseteq G} \lceil \frac{m(H)}{n(H)-1} \rceil,$$

where the maximum is taken over all subgraphs H of G, with $m(H)$ edges and $n(H)$ vertices.

Lemma 6. *For a simple graph G with n vertices and m edges, we have $m \leq 6kn$, or*

$cr_k(G) \geq \frac{1}{64} \cdot \frac{m^3}{n^2k^2}$

Theorem 6. *Let $G = (V, E)$, and let k be a given integer. Let $\{V_1, V_2, \ldots, V_t\}$ be a partition of V and let $H = (V(H), E(H))$ denote the mate of G. If $k \geq a(H)$, then we can construct in polynomial time a k-planar drawing of G with at most*

$$\frac{1}{2}p^2t + 2pq|E(H)| + \frac{1}{2}q^2 \sum_{i=1}^{a(H)} \sum_{x \in V(H)} d_i^2(x)$$

crossings, where $p = max\{|E_{ii}|\} and q = max\{|E_{ij}|\}, i, j = 1, 2, \ldots, t$.

This corollary follows from the above theorem.

Corollary 3. *For $n \geq 1$ $cr_k(K_n) = O(\frac{n^4}{k^2})$*

Theorem 7. *For any graph G on n vertices and m edges,*

$$cr_k(G) \leq \frac{1}{12k^2}(1 - \frac{1}{4k})m^2 + O(\frac{m^2}{kn})$$

The corresponding drawing can be found in polynomial time. For any graph G,

$$cr_k(G) \leq \frac{2cr(G)}{k^{log_2 \frac{8}{3}}} = \frac{2cr(G)}{k^{1.4708\ldots}}$$

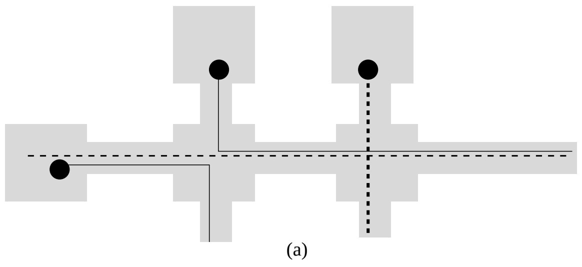
3 The Metro-Line Crossing Minimization Problem

3.1 Motivation

In 1931, graphic designer Harry Beck first proposed that passengers would be more interested in how train lines connect rather than the true geographical layout of stations in a city [3]. This requirement added a new dimension to the problem of minimizing edge crossings in graphs representing maps. His idea became widely accepted in subway stations across the world. Thus, in map drawing, focus came upon how to get from one station to the other and where to change trains rather than on the geographical accuracy of the map. New problems were determined based on this idea. One such problem is the *Metro-Line Crossing Minimization (MLCM) problem* which was introduced by Benkert et al. [5]. Before describing the algorithms that were used to solve it, we describe the MLCM problem.

3.2 Problem background

As we already know, the MLCM problem deals with graph representations of metro-line maps. In the map, each line consists of a sequence of edges and 2 terminals. Each line that traverses a station u has to touch 2 of the sides of u at some points. The stations are of convex, polygonal shapes. The lines can use different number of sides of each station, to enter or leave the station. Two of the most common models that appear in metro-line maps are the 2-side model



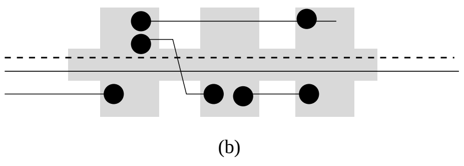


Fig. 7. (a) represents a 4-side model. (b) represents a 2-side model

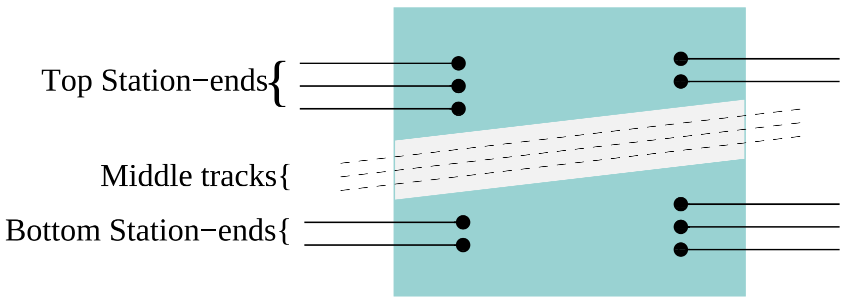


Fig. 8. Station ends and middle tracks

and the 4-side model, as depicted in the Figure 7. A k-side model would consist of stations having k sides each.

An interesting case that arises under the 2-side model [2] and concerns the location of the line terminals at the nodes is the one where the lines that terminate at a station occupy its topmost and bottommost tracks, from here onwards referred to as top and bottom station ends, respectively. The remaining tracks on the left and right side of the station are referred to as middle tracks and are occupied by the lines that pass through the station. Figure 8 illustrates this notion (Solid lines correspond to lines that terminate, whereas the dashed lines correspond to lines that go through the station).

3.3 MLCM problems

Based on the concepts presented in the previous subsecion, now we give definitons of 2 variants of the MLCM problem as given by Bekos et. Al. [2].

(a) *The MLCM problem with terminals at station ends (MLCM-SE)*, where we ask for a drawing of the lines along the edges of G so that (i) all lines terminate at station ends and (ii) the number of crossings among pairs of lines is minimized.

(b) *The MLCM problem with terminals at fixed station ends (MLCM-FixedSE)*, where all lines terminate at station ends and the information whether a line terminates at a top or at a bottom station end in its terminal stations is specified as part of the input. We ask for a drawing of the lines along the edges of G so that the number of crossings among pairs of lines is minimized.

An important part of the MLCM problem concerns the location of the crossings among pairs of lines. If the relative order of two lines changes between two consecutive stations, then the two lines must intersect between these stations, as shown in Figure 7b. We call this an *edge crossing*. On the other hand, a *station crossing* occurs inside a station. In order to avoid the case where a great number of crossings takes place in the interior of a station, which unavoidably leads to cluttered drawings, we seek to avoid station crossings whenever possible. This is an important issue especially when station sizes are small.

As already mentioned, the MLCM problem was first introduced by Benkert et. Al. [5]. They dealt with a simple version of the MLCM problem, the *One-Edge Layout Problem*, which they describe as follows:

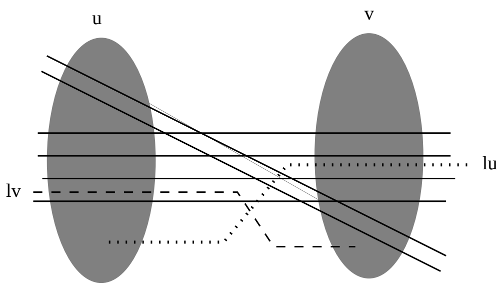


Fig. 9. The lines in L_{uv} are drawn solid.

Given a graph $G = (V, E)$ and an edge $e = \{u, v\} \in E$. Let L_e be the set of lines that use e. We split L_e into three subgroups(see Figure 9): L_{uv} is the set of lines that pass through u and v, i.e., neither u or v is a terminal station. L_u is the set of lines that pass through u and for which v is a terminal station and L_v is the set of lines that pass through v and for which u is a terminal station. We assume that there are no lines that exclusively use the edge $\{u, v\}$ as they could be placed on the topmost or bottommost parts of the station without causing any intersections. Furthermore we assume that the lines for which u is an intermediate station, i.e., $L_{uv} \cup L_u$, enter u in a predefined order S_u . Analogously, we assume that the lines for which v is an intermediate station, i.e., $L_{uv} \cup L_v$, enter v in a predefined order S_v . The task is to find a layout of the lines in L_e or in other words an insertion order such that the number of pairs of intersecting lines is minimized.

Lemma 7. *In any optimal solution for the one-edge layout problem no pair of lines in L_u and no pair of lines in L_v intersects.*

The algorithm they used to solve the one-edge layout problem is a dynamic-programming based algorithm, which runs in $O(n^2)$ time. However, the algorithm fails to address the case for larger graphs.

Bekos et. al. [15] proved that the MLCM-FixedSE problem can be solved $O(|V| + log\Delta \sum_{l \in L} |l|)$ in a case where the underlying network is a tree of degree Δ.

Asquith et. al. [3] extended Bekos et. al.'s work, showing that the MLCM-FixedSE problem is solvable in polynomial time with time complexity $O(|E|^{5/2}/|L|^3)$. They also provided an integer linear programming approach for solving the MLCM-SE problem. Their approach works in 4 steps:
1.For each pair of lines $g, h \in H$ compute all (maximal) common subpaths $\delta_1(g, h), \ldots, \delta_m(g, h)$
2. Each common subpath $\delta(g, h)$is converted into a set of crossing rules C encoding the relations between the terminators of g and h.
3. The positions of these terminators and the number of crossings in G is determined using an integer linear program.
4. The actual line ordering at each vertex is decided.

Bekos et. al. [1], improved the results given by Asquith et. al.Asquith. They gave the following theorems.

Theorem 8. *Given a graph $G = (V, E)$ and a set of lines L on G that terminate at stations of degree 1, the MLCM under the 4-side model can be solved in $O((|E| + |L|^2)|E|)$ time.*

Theorem 9. *Given a graph G = (V, E) and a set of lines L on G that terminate at stations of degree 1, the MLCM problem under the k-side model can be solved in $O((|E| + |L|^2)|E|)$ time.*

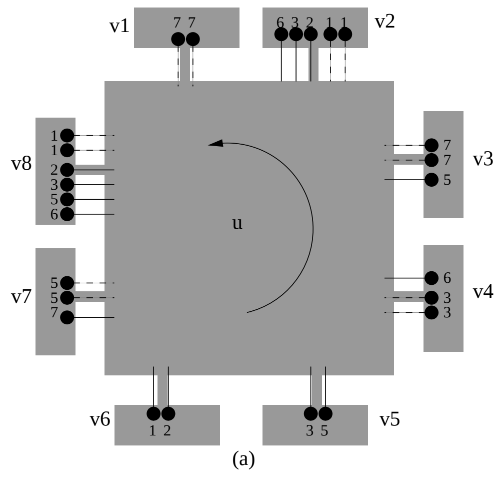
In [2], it was proved that the MLCM-SE problem on a path is NP-complete, even in the case where the underlying network is a path. Here, we omit the details. They first show that, given a positive integer $c \in Z^+$, the problem of finding a solution of the MLCM-SE problem on a path with total number of crossings no more than c, is NP-complete. Membership in NP follows from the fact that a nondeterministic algorithm needs only to guess an ordering of the lines at the left and the right side of each station and then to check whether the total number of crossings of the implied solution is no more than c, which can be clearly done in polynomial time. They then show that Masuda et. Al.'s *fixed linear crossing number problem*, which was proved to be NP-Hard [12], is reducible to the MLCM-SE problem on a path using examples. Then they in turn show that the latter is reducible to the fixed linear crossing number problem. As a result, they show that the crossings in the examples from the 2 problems bear a one-to-one correspondence, hence completing the proof.

We now discuss the algorithm (whose complexity is given in Theorems 8 and 9) by Bekos et. Al. [2] that solves the MLCM problem under the k-side model. It is assumed that there exists no restriction on the location of the line terminals at the nodes. In addition to that, it is assumed that the lines that terminate at a station occupy its top and bottom station ends. Since the order of the stations is fixed as part of the input of the problem, the only remaining choice is whether each line terminates at the top or at the bottom station end in its terminal stations. (Referring to figure 7 stations of degree are referred to as *terminal stations*. Stations of degree greater than one are referred to as *internal stations*. Their algorithm works for the 4-side model and can be extended, by recursion, for the k-side model. As previously mentioned, in a 4-side model each station is represented as a rectangle and tracks are permitted to all four sides of each station. It is further assumed that an internal station always exists within the underlying network, otherwise the problem can be solved trivially.

The basic idea of their algorithm is to decompose the underlying network by removing an arbitrary edge out of the edges that connect two internal stations (and, consequently, appropriately partitioning the set of lines that traverse this edge), then recursively solve the subproblem and, finally, derive a solution of the initial problem by i) re-inserting the removed edge and ii) connecting the partitioned lines along the re-inserted edge.

The base of the recursion. The base of the recursion corresponds to the case of a graph G_B consisting of an internal station u and a number of terminal stations, say $v_1, v_2, \ldots, v_f$, incident to u, each of which has only one side with terminals (see Figure 10). To cope with this case, all lines that have exactly the same terminals are grouped into a single line, which is referred to as bundle. The lines in a bundle will be drawn in a uniform fashion, i.e., occupying consecutive tracks at their common stations. In an optimal solution, a bundle can be safely replaced by its corresponding lines without affecting the optimality of the solution. In Figure 10b, lines belonging to the same bundle have been drawn with the same type of non-solid line. The single lines are referred to as bundles, too, in order to maintain a uniform terminology (refer to the solid lines of Figure 10)). Then, the number of bundles of each terminal station is bounded by the degree of the internal station u.

The numbering step. In order to route the bundles along the edges of G_B, *Euler tour numbering* technique is used. Let v be a terminal station of G_B. Then, the



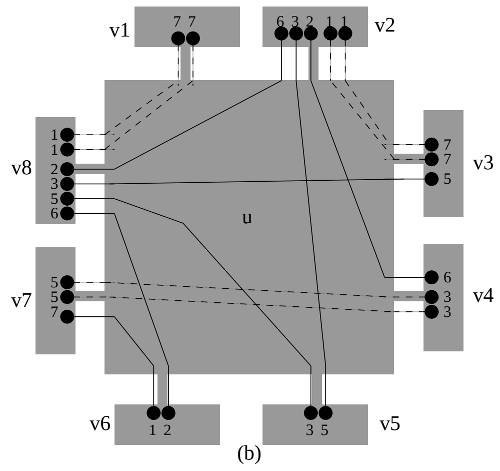


Fig. 10. Illustration of the base of the recursion

Euler tour numbering of the terminal stations $v_1, v_2, \ldots, v_f$ of G_B with respect to v is a function $ETN_v : \{v_1, v_2, \ldots, v_f\} \rightarrow \{0, 1, \ldots, f-1\}$. More precisely, all terminal stations of G_B are numbered according to the order of first appearance when moving clockwise along the external face of G_B starting from station v, which is assigned the value zero. Note that such a numbering is unique with respect to v. This numbering is referred to as the Euler tour numbering starting from station v or simply as ETN_v. In Figure 10b, the number next to each line terminal at each terminal station v_i corresponds to the ETN_{v_i} of its destination, $i = 1, 2 \ldots 8$. By using the following formula, computation of only one numbering is enough in order to derive the corresponding Euler tour numberings from any other terminal station v of G_B,

$ETN_v(w) = (ETN_v(w)ETN_v(v))\mathrm{mod} f$.

The sorting step. We first sort the bundles at each terminal station v based on the Euler tour numbering starting from v (i.e., ETN_v) of their destinations in ascending order and we place them so that they appear in counterclockwise order around the internal station u. We denote by $BND(v)$ the ordered set of bundles of each terminal station v. Then, we pass these bundles from each terminal station to the internal station u along their common edge without introducing any crossings. This implies an ordering of the bundles at each side of the internal station u. To complete the routing procedure, it remains to connect equal bundles in the interior of the internal station u, which may imply crossings (see Figure 10b). Note that only station crossings that cannot be avoided are created in this manner since the underlying network is planar. The Euler tour numbering implies that no unnecessary edge crossings occur. Therefore the optimality of the solution follows trivially.

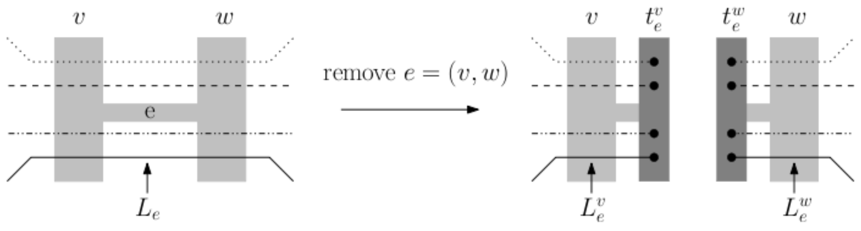


Fig. 11. Illustration of the removal of an edge that connects two internal stations

The Recursive Algorithm. If the input graph is not connected we can separately solve the MLCM problem on each of the connected components of G and the lines induced by each of these components. Thus, in the rest of the algorithm, it is assumed that the input graph is connected. Let $e = (v, w)$ be an edge which connects two internal stations v and w of the underlying network. If no such edge exists, then the problem can be solved by employing the algorithm of the base of the recursion.

Let L_e be the set of lines that traverse e. Any line $l \in L_e$ originates from a terminal station, passes through a sequence of edges, then enters station v, traverses edge e, leaves station w and, finally, passes through a second sequence of edges until it terminates at another terminal station. Since each line consists of a sequence of edges, L_e can be written in the form $\{l \in L | l = \pi e \pi'\}$. We proceed by removing edge e from the underlying network and by inserting two new terminal stations t_e^v and t_e^w incident to the stations v and w, respectively (see the dark-gray colored stations of the right drawing of Figure 11). Let $G^* =$

$(V \cup \{t_e^v, t_e^w\}, (E\{e\}) \cup \{(v, t_e^v), (t_e^w, w)\})$ be the new underlying network obtained in this manner.

Since the edge e has been removed from the underlying network, the lines of L_e cannot traverse e anymore. So, they are forced to terminate at t_e^v and t_e^w, as it is depicted in the right drawing of Figure 11. More precisely, let:

$L_e^v = \{\pi(v, t_e^v) | \pi e \pi' \in L_e\}$

$L_e^w = \{(t_e^w, w)\pi' | \pi e \pi' \in L_e\}$

Then, the new set of lines that is obtained after the removal of the edge e is $L^* = (L - L_e) \cup (L_e^v \cup L_e^w)$. We proceed by we recursively solving the MLCM problem on (G^*, L^*). The recursion will lead to a solution of (G^*, L^*). Part of the solution consists of two ordered sets of bundles $BND(t_e^v)$ and $BND(t_e^w)$ at each of the terminal stations t_e^v and t_e^w, respectively. Recall that, in the base of the recursion, all lines in a bundle have exactly the same terminals. In the recursive step, a bundle actually corresponds to a set of lines (with the same terminals) whose relative positions cannot be determined. In order to construct a solution of (G, L), we first have to restore the removed edge e and to remove the terminal stations t_e^v and t_e^w. The bundles $BND(t_e^v)$ and $BND(t_e^w)$ of t_e^v and t_e^w have also to be connected appropriately along the edge e. Note that the order of the bundles of t_e^v and t_e^w is equal to those of v and w, because of the base of the recursion. Therefore, the removal of t_e^v and t_e^w will not produce unnecessary crossings.

We now describe the procedure of connecting the ordered bundle sets $BND(t_e^v)$ and $BND(t_e^w)$ along edge e. We say that a bundle is of size s if it contains exactly s lines. Two bundles are equal if they contain the same set of lines, i.e., the parts of the lines that each bundle contains correspond to the same set of lines. First, all equal bundles are connected. Let $b \in BND(t_e^v)$ and $b' \in BND(t_e^w)$ be two equal bundles. The connection of b and b' will result into a new bundle which contains the lines of b (or equivalently of b') and its terminals are the terminals of b and b' that do not participate in the connection. Note that a bundle is specified as a set of lines and a pair of stations, that correspond to its terminals. When the connection of b and b' is completed, both b and b' are removed from $BND(t_e^v)$ and $BND(t_e^w)$.

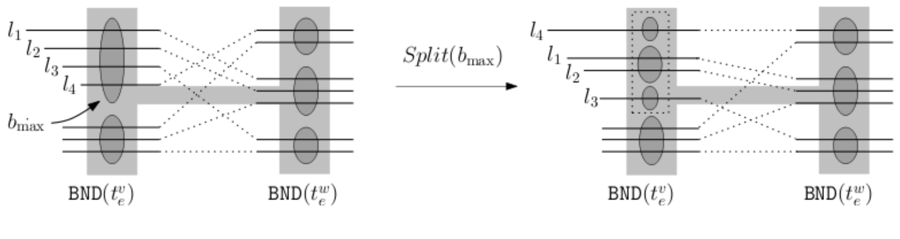


Fig. 12. Splitting the largest bundle. The dotted lines just illustrate which connections have to be made.

If both $BND(t_e^v)$ and $BND(t_e^w)$ are empty, all bundles are connected. In the case where they still contain bundles, we determine the largest bundle, say b_{max} of $BND(t_e^v) \cup BND(t_e^w)$. Without loss of generality, we assume that $b_{max} \in BND(t_e^v)$ (see the left drawing of Figure 12). Since b_{max} is the largest bundle among the bundles of $BND(t_e^v) \cup BND(t_e^w)$ and all equal bundles have been removed from both $BND(t_e^v)$ and $BND(t_e^w)$, b_{max} contains at least two lines that belong to different bundles of $BND(t_e^w)$. So, it can be split into smaller bundles, each of which contains a set of lines belonging to the same bundle in $BND(t_e^w)$(see the right drawing of Figure 12). Also, the order of the new bundles in $BND(t_e^v)$ must follow the order of the corresponding bundles in $BND(t_e^w)$ in order to avoid unnecessary crossings (refer to the order of the bundles within the dotted rectangle of Figure 12). In particular, the information that a bundle was split is propagated to all stations that this bundle traverses, i.e., splitting a bundle is not a local procedure that takes place along a single edge but it requires greater effort. Note that no crossings among lines of b_{max} occur, when b_{max} is split. In addition, the crossings between lines of b_{max} and bundles in $BND(t_e^w)$ cannot be avoided, which proves the correctness of the algorithm.

These two steps (i.e., connecting equal bundles and splitting the largest bundle) are repeated until both $BND(t_e^v)$ and $BND(t_e^w)$ are empty. Since the largest bundle is always split into smaller ones, it is guaranteed that the algorithm for the connection of the bundles along the edge e will eventually terminate.

This sums up the algorithm for solving the MLCM-SE problem for the k-side model.

4 Scope for future work

The MLCM problem has only been introduced lately. MLCM models dicussed in this paper offer new types of graphs models to work on, which contain several restrictions and constraints with respect to the ordering of the vertices and the placements of the edges. We thus suggest exploration in permutation techniques for implementation for MLCM graph models in order to find the minimum number of edge crossing. Because of the restrictions in such graphs, they will have a smaller permutation search field as compared to other, general, graphs. Thus some useful results could be gained by applying permutation techniques in this are.

We also propose a result that we established experimentally, relating to edge crossing in a bipartite graph. The result is as follows,

Consider a bipartite graph G which contains equal number of vertices (say k) in both of its vertex sets and let there be no multi-edges. Then G will have an edge crossing if it contains more than $4k-4$ *edges among its vertices.*

Figure 13 shows such a bipartite graph, where $k = 3$. We see that at most 2 vertices in a set can have degree 3, while the remaining single vertex can have

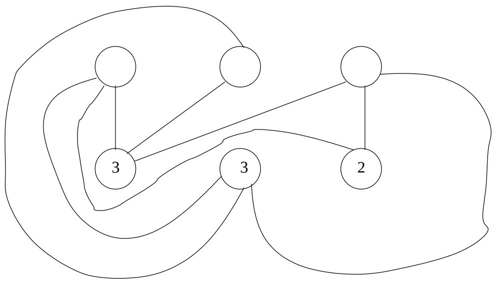


Fig. 13. A bipartite graph G with $4k - 4$ edges. The numbers represent the degrees of the vertices indicated (here $k = 3$)

at most a degree of 2, for the graph G to have no edge crossings. If $k = 4$, we see that at most 2 vertices in a set can have degree 4, while the remaining vertices can have at most a degree of 2 each, for the graph G to have no edge crossings. The same is seen, for all k. Thus the maximun number of edges that can exist in the bipartite graph G is, $2 \times k + (k - 2) \times 2$ which gives $4k - 4$.

For the purpose of edge crossing minimization, one could identify such a bipartite graph(s) in any graph (whose number of edge crossings we wish to minimize), and then check the number of edges in the bipartite graph(s). If the number is $\leq 4k-4$, then the bipartite graph can definitely be transformed into a drawing with no edge crossings. Consequently, the transformed biparite graph(s) can be replaced with the former in the main graph, optimizing the number of crossings in the main graph.

5 Conclusion

Edge crossing minimization has long been a field of extensive study and research. In this survey paper, we discussed various important results and algorithms that have been carried out over the last few decades. We particularly focused on the edge crossing minimization techniques and results for different types of graphs, including book drawings, hierarchical graphs, and symmetric graphs. A new breed of graph models that have emerged from recent study are 2-side models, 4-side models, and the general k-side models, which are used to depict stations and tracks in metro maps. Some interesting crossing minimization techniques that are applied to such graph models are analysed and compared in this paper. Finally, we also give some future directions in the study of the metro-line crossing minimization problem. We also proposed an important result for bipartite graphs which we hope would prove instrumental in designing efficient edge crossing minimization algorithms.

References

1. Argyriou, E., Bekos, M.A., Kaufmann, M., Symvonis, A.: Two polynomial time algorithms for the metro-line crossing minimization problem. In: 16th International Symposium on Graph Drawin (GD'08). vol. 5417, pp. 336–347 (2008)
2. Argyriou, E., Bekos, M.A., Kaufmann, M., Symvonis, A.: On metro-line crossing minimization. Journal of Graph Algorithms and Applications 14(1), 75–96 (2010)
3. Asquith, M., Gudmundsson, J., Merrick, D.: An *ilp* for the metro-linecrossing problem. In: Fourteenth Computing: The Australasian Theory Symposium (CATS'08). vol. 77, pp. 49–56 (2008)
4. Baur, M., Brandes, U.: Crossing reduction in circuit layouts. In: 30th Int. Workshop Graph-Theoretic Concepts in Computer-Science. vol. 3353, pp. 332–343 (2006)
5. Benkert, M., Nollenburg, M., Uno, T., Wolff, A.: Minimizing intra-edge crossings in wiring diagrams and public transport maps. In: 14th International Symposium on Graph Drawing. vol. 4372, pp. 270–281 (2006)
6. Buchheim, C., Hong, S.: Crossing minimisation for symmetries. Theory of Computing Systems 38, 293–311 (2005)
7. Cimikowski, R.: Algorithms for the fixed linear crossing number problem. Discrete Applied Mathematics pp. 93–115 (2002)
8. Eades, P., Lin, X.: Spring algorithms and symmetry. Theoretical Computer Science 2(240), 379–405 (2000)
9. He, H., Sykora, O.: New circular drawing algorithms. In: ITAT'2004 (2004)
10. He, H., Sykora, O.: Crossing minimisation heuristics for 2-page drawings. Electronic notes in discrete mathematics 22, 527–534 (2005)
11. Masuda, S., Nakajima, K., Kashiwabara, T., Fujisawa, T.: On the np-completeness of a computer network. In: IEEE Intl. Symposium on Circuits and Systems. pp. 292–295 (1987)
12. Masuda, S., Nakajima, K., Kashiwabara, T., Fujisawa, T.: Crossing minimization in linear embeddings of graphs. In: Proceedings of the Institution of Electrical Engineers. vol. 39, pp. 124–127 (1990)
13. Nash-Williams, J.: Edge disjoint spanning trees of finite graphs. In: J. London Math Society. vol. 36, pp. 445–450 (1961)
14. Nicholson, T.A.J.: Permutation procedure for minimising the number of crossings in a network. In: Proceedings of the Institution of Electrical Engineers. vol. 115, pp. 21–26 (1968)
15. Potika, K., Bekos, M.A., Kaufmann, M., Symvonis, A.: Line crossing minimization on metro maps. In: 15th International Symposium on Graph Drawin (GD'07). vol. 4875, pp. 231–242 (2007)
16. Shahrokhi, F., Sykora, O., Szekely, L.A., Vrt'o, I.: On k-planar crossing numbers. Discrete Applied Mathematics pp. 1106–1115 (2007)
17. Weixiang, S., Jingwei, H.: Edge crossing minimization algorithm for hierarchical graphs based on genetic algorithms. Wuhan University Journal of Natural Sciences 6(1-2), 555–559 (2001)